12

University of Southern California

Two Frameworks for Integrating Knowledge in Induction

Paul S. Rosenbloom and Benjamin D. Smith
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Haym Hirsh
Dept. of Computer Science
Hill Center Busch Campus
Rutger University

William W. Cohen
AT&T Bell Laboratories

1 28 16

Two Frameworks for Integrating Knowledge in Induction

Paul S. Rosenbloom and Benjamin D. Smith
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Haym Hirsh
Dept. of Computer Science
Hill Center Busch Campus
Rutger University

William W. Cohen
AT&T Bell Laboratories
600 Mountain Ave
Murray Hill, NJ 07974

September 1993
ISI/RR-93- 361

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1, 1993	3. REPORT TYPE AND DATES COVERED Technical

4. TITLE AND SUBTITLE Two Frameworks for Integrating Knowledge in Induction	5. FUNDING NUMBERS NASA-NCC 2-538

6. AUTHOR(S)

Paul S. Rosenbloom	Haym Hirsh	William W. Cohen	Benjamin D. Smith
Information Sciences Institute	Dept. of Computer Science	AT&T Bell Laboratories	Information Sciences
University of Southern California	Hill Center, Busch Campus	600 Mountain Avenue	Dept. of Comp. Sci.
4676 Admiralty Way	Rutgers University	Murray Hill. NJ 07974	Univ. of So. CA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATON REPORT NUMBER RR361
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) NASA	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT UNCLASSIFIED/UNLIMITED	12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The use of knowledge in inductive learning is critical for improving the quality of the concept definitions generated, reducing the number of examples required in order to learn effective concept definitions, and reducing the computation needed to find good concept definitions. Relevant knowledge may come in many forms (such as examples, descriptions, advice, and constraints) and from many sources (such as books, teachers, databases, and scientific instruments). How to extract the relevant knowledge from this plethora of possibilities, and then to integrate it together so as to appropriately affect the induction process is perhaps the key issue at this point in inductive learning. Here we focus on the integration part of this problem; that is, how induction algorithms can, and do, utilize a range of extracted knowledge. Preliminary work on a transformational framework for defining knowledge-intensive inductive algorithms out of relatively knowledge-free algorithms is described, as is a more tentative problem-space framework that attempts to cover all induction algorithms within a single general approach. These frameworks help to organize what is known about current knowledge-intensive induction algorithms, and to point towards new algorithms.

14. SUBJECT TERMS	15. NUMBER OF PAGES 10
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Two Frameworks for Integrating Knowledge in Induction

Paul S. Rosenbloom
Information Sciences Institute &
Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292

Haym Hirsh
Department of Computer Science
Hill Center, Busch Campus
Rutgers University
New Brunswick, NJ 08903

William W. Cohen
AT&T Bell Laboratories
600 Mountain Avenue
Room 3C-412A
Murray Hill, NJ 07974

Benjamin D. Smith
Information Sciences Institute &
Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292

September 1, 1993

Abstract

The use of knowledge in inductive learning is critical for improving the quality of the concept definitions generated, reducing the number of examples required in order to learn effective concept definitions, and reducing the computation needed to find good concept definitions. Relevant knowledge may come in many forms (such as examples, descriptions, advice, and constraints) and from many sources (such as books, teachers, databases, and scientific instruments). How to extract the relevant knowledge from this plethora of possibilities, and then to integrate it together so as to appropriately affect the induction process is perhaps the key issue at this point in inductive learning. Here we focus on the integration part of this problem; that is, how induction algorithms can, and do, utilize a range of extracted knowledge. Preliminary work on a transformational framework for defining knowledge-intensive inductive algorithms out of relatively knowledge-free algorithms is described, as is a more tentative problem-space framework that attempts to cover all induction algorithms within a single general approach. These frameworks help to organize what is known about current knowledge-intensive induction algorithms, and to point towards new algorithms.

Introduction

Inductive learning is a process whereby a definition of a concept is derived from a set of positive, and sometimes negative, examples of the concept. Key issues in inductive learning are the accuracy of the resulting definition – that is, the error rate it yields in classifying new examples – and the resources required to generate the definition (in terms of the number of examples and/or the amount of time and space needed).

The single most promising route towards reducing both the error rate and resource usage of inductive learning is to utilize whatever additional knowledge is available beyond the examples;

that is, to convert induction from a weak to a strong method. However, to do this, the relevant knowledge must first be extracted from the sources in which it exists, such as books, teachers, databases, and scientific instruments. This extracted knowledge must then be integrated together for use by the induction process, in whatever form is appropriate – examples, descriptions, advice, constraints, or anything else. Here we focus on the integration task (extraction involves potentially everything from vision to natural language understanding, and more). Our goal is to begin the process of deriving principles for how knowledge-intensive induction algorithms both do now, and can in the future, provide such integration. The hope is that this will lead to both a useful descriptive framework for organizing existing approaches, as well as a prescriptive framework for generating new approaches that go beyond the existing ones.

We'll make this beginning by presenting two partial frameworks for knowledge integration in induction, along with implications drawn so far by applying them to four recently proposed knowledge-intensive induction algorithms. The focus here is specifically on knowledge integration for induction, rather than on the broader issue of knowledge integration in general, in the hope that the extra structure provided by the induction problem will lead to more powerful integration strategies than have been proposed for the general case. The more developed of the two knowledge-integration frameworks – and thus the one emphasized in this paper – is the *transformational* framework. It describes how knowledge-intensive induction algorithms are, and can be, derived by transforming traditional learning methods. The more tentative *problem-space* framework attempts to go beyond the transformational framework to the more difficult task of characterizing the fundamental components of all induction algorithms, whether knowledge-intensive or not. This framework is covered only briefly here as an intriguing possibility for the future.

The Transformational Framework

An induction algorithm can be viewed abstractly as a black box with one output port for the concept description and one or more input ports. A minimal induction algorithm has just one input port, for training data, with all other information being hardwired into the algorithm as a fixed *bias* (Mitchell, 1980). The only way such an algorithm can use additional knowledge – other than by reprogramming – is to find some way of recoding the knowledge as pseudo-examples. For example, Quinlan describes how knowledge about type restrictions on the arguments of predicates could conceivably be used indirectly by the FOIL algorithm through the generation of pseudo-negative examples that cause FOIL to eliminate candidate hypotheses that would violate the type restrictions (Quinlan, 1990).

Most induction algorithms actually do provide some additional input ports that allow explicit provision of other types of information; that is, of knowledge beyond what is embodied in the examples. For example: the candidate elimination algorithm provides an input port for information concerning the partial ordering that defines the initial hypothesis space (Mitchell, 1977); FOIL provides an input port for the set of relations that can be used in candidate hypotheses (Quinlan, 1990); backpropagation provides input ports for learning-rule parameters, the network structure, and the initial connection weights (Rumelhart, Hinton, & Williams, 1986); and Bayesian learning algorithms provide an input port for prior probabilities (Buntine, 1991). Such ports expand the types of information that can be utilized at induction time, but still provide a very limited means for incorporating the full range of knowledge that may be available.

The transformational framework starts with the basic notion of black boxes and ports, as described above, and views knowledge-intensive induction algorithms as the composition of a core, usually knowledge-lean, algorithm plus a set of transformations. Although not all knowledge-intensive algorithms can be viewed in this fashion, when they can, the results can be quite informative. The four knowledge-intensive algorithms covered in the next section do all fit this framework, and are based on three distinct core algorithms – candidate elimination, FOIL, and backpropagation – and on two general types of transformations to these core algorithms:

- A *reformulation* transformation modifies the core algorithm so that its ports can handle a wider range of inputs, either by generalizing its existing ports or by adding new ones. A simple example of a reformulation is the modification of a decision-tree learner to allow it to accept a task-specific split criterion from an input port, rather than always using the same built-in criterion. A more sophisticated example of a reformulation is IVSM's derivation from the candidate elimination algorithm by converting its examples input port to take a more expressive class of inputs (i.e., version spaces) (Hirsh, 1990).

- A *preprocessor* transformation adds to the core algorithm a preprocessor that takes a form of input beyond what can be fed directly into the core algorithm's input ports, and translates this broader input into something that one or more of the core input ports can understand. Quinlan's suggestion of using type constraints to create pseudo-negative examples is exactly a proposal for a preprocessor transformation. The preprocessor would have an input port that can accept type constraints, and would produce negative examples that can be fed into FOIL's existing input port. The combination of FOIL and this preprocessor thus defines a new learning algorithm that can take as input not only examples and relations, but also type constraints.

Examples

Much recent work on induction algorithms has focused on enhancing their ability to utilize additional knowledge during induction, and thus there are many learning systems we could consider. A full survey of such algorithms is beyond the scope of this paper, so we will focus here on just four recent knowledge-intensive algorithms, each of which provides the ability to utilize EBL-like domain theories, plus possibly some other forms of knowledge:

- IVSM has the ability to utilize EBL-like domain theories plus models of bounded inconsistency (Hirsh, 1990).

- FOCL has the ability to utilize (possibly partial) EBL-like domain theories plus constraints on predicate arguments (Pazzani & Kibler, 1992).

- GRENDEL has the ability to specify the hypothesis space via a formal grammar – which can include an EBL-like domain theory – plus some simple ordering information (Cohen, 1992).

- KBANN is a neural network algorithm that has the ability to utilize an EBL-like domain theory (Towell, Shavlik, & Noordewier, 1990).

The remainder of this section considers these four systems in more detail.

IVSM is based on the candidate-elimination algorithm (CEA). It is derived by a reformulation of the CEA so that instead of basing its inner loop on the process of updating a version space with respect to a single example, it now updates the version space with respect to a second version space (by intersecting the two version spaces). This reformulation generalizes the CEA's examples input port so that it now accepts version spaces. In addition to this core reformulation transformation, IVSM also uses three distinct preprocessor transformations that are enabled by this reformulated input port. One preprocessor allows IVSM to emulate the CEA by taking examples and converting them into version spaces. A second preprocessor creates version spaces from combinations of examples and EBL-style domain theories. A third preprocessor creates version spaces from combinations of examples and a model of bounded inconsistency. When IVSM is combined with any one of these preprocessors, it actually yields a new induction algorithm: IVSM-CEA, IVSM-EBL, or IVSM-BI.

FOCL is based on FOIL. FOIL uses the information provided on its relations input port to determine what modifications to consider making to the current candidate hypothesis. Essentially, it considers adding the various relations – as instantiated with a mixture of old and new variables – to the current clause of the hypothesis, and uses an information-theoretic measure to determine which possibility is (locally) best. FOCL reformulates this possibility-generation strategy in two ways. First, it increases the set of possibilities by considering adding combinations of relations in a single step, rather than just individual relations. Second, it decreases the set of possibilities by eliminating those that violate constraints on the arguments of the relations (such as type and uniqueness restrictions). The first reformulation supports the addition of an input port for (possibly partial) EBL-like domain theories; the combinations of relations that occur in the condition sides of these rules form the basis for the relation combinations proposed in the reformulated algorithm. The second reformulation supports the addition of an input port for type and uniqueness constraints on the arguments of the relations that are proposed. This new port directly supports knowledge that FOIL would have needed a preprocessor to use.

GRENDEL is also based on FOIL. The core transformation made in developing GRENDEL is also a reformulation of FOIL's generation strategy for possible modifications to the current candidate hypothesis. However, GRENDEL's reformulation is both more radical and more general. GRENDEL generates possibilities by consulting generation rules specified in a context-free grammar. This supports broadening FOIL's input port from one that can take a list of relations to one that can handle a list of context-free grammar rules. A second smaller reformulation allows the processing of possibilities to be selectively deferred, and supports the addition of a second input port to specify this simple ordering information. The remainder of the GRENDEL story is much like IVSM. The generalized input port facilitates the creation of a number of preprocessors that can accept a variety of types of input. This input is then translated into grammar rules that can be fed to this new port. These preprocessors allow GRENDEL to accept the kinds of input utilized by (among others) EBL, FOIL, and FOCL, and thus to emulate these other algorithms. As with IVSM, there is a base GRENDEL algorithm which takes grammars as inputs, and then there are a number of other induction algorithms that are based on GRENDEL, such as GRENDEL-EBL and GRENDEL-FOIL, which are derived from it by adding specific preprocessors.

Finally, KBANN is based on backpropagation. It adds a preprocessor that takes as input an EBL-like domain theory, plus a list of environmental features not covered by the theory, and translates this knowledge into a form that can be fed into backpropagation's initial network topology and initial network weights ports. It leaves backpropagation's remaining input ports – such as its learning-rule parameters – intact.

Analyzing Input Ports in the Transformational Framework

The transformational framework makes it possible to examine knowledge-intensive learners in more detail, by studying the set of input ports provided by the resulting algorithms, what kind of knowledge they can accept, and what key properties they possess (or fail to possess). Although we are still in the process of identifying what the key properties are for input ports, the list already includes at least two that seem critical.

- The *additivity* of an input port is determined by its ability to accept multiple independent fragments of knowledge at that port. Additivity is important because additive ports can serve directly as integrators for arbitrary amounts of knowledge of the types that they can accept. The prototypical example of an additive input port is the example port in standard induction algorithms. It can accept arbitrary amounts of new information, and combine it straightforwardly with whatever else the system knows. A classical example of a non-additive port is the learning-rate parameter in backpropagation. If more information is available, how should it be combined with what is already known? Must the old information simply be eliminated, and replaced by the new, or should the two values be averaged, or should something else happen?

  For additive ports, the way in which inputs are combined usually depends on their interpretation. Examples can be viewed as constraints on the behavior of the concept being learned, so they are usually combined via an intersection operator. Other types of information might be combined via different operators, such as union or average.

- The *ease of use* of an input port is determined by how easy it is to express knowledge in the language provided by the port. Bayesian priors are a classic case of a difficult-to-use input port, with this difficulty most likely being the single biggest stumbling block in using Bayesian approaches to learning. Sometimes preprocessors can be added to make a port easier to use; however, the port's basic ease of use will still affect how easy it is to write the preprocessors. A good example of such a preprocessor for Bayesian priors is the use of the *minimum description length* principle, which, while it can be viewed as a Bayesian approach, replaces the task of assigning a prior probability to every concept with the arguably simpler task of choosing an encoding scheme (Quinlan & Rivest, 1989).

To illustrate these two properties of knowledge-intensive induction methods, as viewed from the transformational framework, we return to the four algorithms discussed above.

The core IVSM algorithm has two input ports, one for the partial-order information on which the version spaces are based and one for a collection of version spaces. The partial-order port is additive because it can handle an arbitrary number of elements plus ordering relations among them. It is also easy to use, but only for the narrow purpose of identifying (possibly parts of) candidate hypotheses and generality relationships among them. The version-space port is also an additive port – as with the traditional example input port, it can accept an unbounded set of inputs, and combine them (via version-space intersection) with what is already known. Its ease of use is intermediate between that provided by example ports at the low end (at least if they are being used for anything other than just examples) and languages like GRENDEL's grammars at the high end. When IVSM's preprocessors are considered, there are three new input ports, all of which are additive and relatively easy to use (for the restricted uses for which they are intended).

One idea that is directly suggested by this analysis of IVSM is that there is no reason its three distinct preprocessors couldn't all be used simultaneously. Because they all output version spaces, and the version-space port is additive, it should be possible to intermingle information based on examples, domain theories, and bounded inconsistency (thus effectively creating a new algorithm that subsumes the three existing ones).

FOCL's three input ports – for examples, (possibly partial) domain theories, and argument constraints – are all additive, as they can all accept arbitrary amounts of knowledge of their chosen input types. They are also all easy to use for their intended purposes, but difficult to use for other purposes.

GRENDEL's three input ports accept examples, grammars, and ordering information (information about what portions of the hypothesis space should be tried first). Regarding ease of use, the example port has the standard properties; the ordering port is similar to a Bayesian-priors port but likely to be somewhat easier to use because it is much less demanding; and the grammar port is relatively easy to use for most purposes. The example and ordering ports are both additive; however the grammar port is only semi-additive, in that the grammars are closed under union, but not under intersection. Thus the additivity of the grammar port depends on the way in which grammars are used. If a grammar is used as a suggestion as to which hypotheses are most likely – as when grammars are used to encode a *domain theory* – then grammars can be easily combined with a union operator. However, when grammars are used as constraints on the hypothesis space, it is impossible to generate a separate grammar for each constraint and then integrate the constraints by intersecting the grammars (as IVSM would intersect its version spaces).

KBANN's three input ports accept examples, domain theories, and environmental features. The examples port is much like any other examples port – it is additive and easy to use for its intended purpose (but difficult to use for other purposes). The domain theory port is additive and easy to use. The environmental-features port is like the examples port, being additive and easy to use for very limited purposes.

Implications of the Transformational Framework

Pulling back up now from these detailed analyses to look at the picture more globally, several general implications can be discerned. The first implication is that multiple pieces of knowledge can be combined in three distinct fashions. The first approach feeds the knowledge into multiple of the core algorithm's input ports, and depends on the structure of the core algorithm to perform the integration. For example, KBANN integrates a domain theory with examples by feeding the domain theory to the core network topology and weight ports, while feeding examples directly to the core examples port. The core algorithm – that is, backpropagation – then combines this knowledge during its normal processing. The second approach utilizes a multi-ported preprocessor that integrates the knowledge provided to its input ports in the process of generating input for the core algorithm. One example is GRENDEL-FOCL's emulation of FOCL via a preprocessor that combines knowledge from all of FOCL's input ports (except for the examples port) in the process of converting this knowledge into a single grammar for use by GRENDEL. A second example is IVSM-EBL's use of a preprocessor to integrate knowledge from its examples and domain-theory ports in the process of generating version spaces for the core IVSM algorithm. The third integration approach is to utilize an additive port that can integrate across multiple pieces of knowledge sent to a single port. IVSM is a good example of this, as its version-space port is an effective additive input port.

The second implication is that the insights underlying different knowledge-intensive algorithms can often be transferred or combined in useful ways. In cases where two knowledge-intensive algorithms are based on the same core algorithm, and where they have transformed the core algorithm in different ways, it should be possible to combine many of the transformations without a great deal of difficulty. For example, GRENDEL's generalization of FOIL's relations port to accept grammars could be combined with FOCL's techniques for pruning hypotheses using typing constraints. It would be an interesting question to see whether this approach would have any advantages over using a preprocessor to incorporate all of FOCL's knowledge into a GRENDEL grammar, as in GRENDEL-FOCL.

In cases where the core algorithms are different, transfer of a more abstract sort can still occur. For example, IVSM's additivity based on version-space intersection leads to asking whether GRENDEL's grammars could support a comparable operation: the answer is no, since context-free grammars are not closed under intersection. This also suggests the new research topic of modifying GRENDEL so that it is more additive. For example, since the intersection of a context-free language and a regular language is a context-free language, it might be possible to create a new version of GRENDEL that has an additive port for regular languages in addition to the existing (non-additive) port for context-free languages.

The third implication is that additional effort would be usefully spent looking at how the two general classes of transformations could be applied to further aspects of existing algorithms, both those considered here as well as others.

Beyond the Transformational Framework

The transformational framework is somewhat unsatisfying for several reasons: it does not apply to all knowledge-intensive learners; it does not apply to knowledge-weak learners (which actually do achieve some forms of knowledge-integration even in simply being able to accept varying numbers of examples and learn from them); and it doesn't say much about how to merge the insights across knowledge-intensive algorithms that have different core algorithms. Our continuing work attempts to go beyond the transformational framework by developing a *problem space* framework that attempts to identify the core functionalities that underly all induction algorithms, and then to understand how all of the knowledge utilized by a learner – examples, domain theories, etc. – is integrated together via its mapping on to these functionalities. In terms of the transformational framework, the goal here can be expressed as finding a single black box and set of input ports that conceptually lie at the heart of all induction algorithms.

The problem-space framework is organized around the concept of the space of candidate hypotheses, thereby continuing the existing line of analyses that have viewed induction as search (Simon & Lea, 1974; Mitchell, 1979; Rosenbloom, 1988). In this framework the role of knowledge is first off to specify, constrain, and order the elements – that is, the states – of this space. In the four algorithms we have focused on here, specification of the states in the space occurs rather directly via GRENDEL's grammar port, FOCL's relations port, and IVSM's partial-order port. Constraints on the set of states considered are provided by IVSM's version spaces and FOCL's argument constraints. Ordering information about the states is provided by GRENDEL's ordering port and GRENDEL's and FOCL's examples ports (though rather indirectly, through their information-theoretic measures). However, none of the four systems allows direct statement of all three types of knowledge. GRENDEL comes the closest, though it requires all constraints to be stated indirectly in terms of what can be generated via the grammar. KBANN is the furthest away, as it cannot accept direct statement of any of these types of knowledge. It does however accept some such information indirectly; for example, its domain theory (plus information about additional domain features) indirectly determines what can and cannot be in the hypothesis space, by determining the network topology.

The remaining use of knowledge in this framework is to provide method-specific knowledge about how to search the space of hypotheses. IVSM is at one extreme, in that it makes no use of such knowledge – it always maintains a representation of all hypotheses that are consistent with all of the knowledge available so far. FOCL, GRENDEL, and KBANN all utilize greedy search algorithms. FOCL uses its relation and domain-theory ports to generate candidate changes at each step, its argument-constraint port to eliminate candidates, and its examples port to order the candidates (via its information-theoretic measure). GRENDEL uses the detailed structure of its grammar rules to generate the candidates at each step – two grammars that generate the same terminal language could lead to different greedy searches if they are specified in terms of different sets of rules. It also uses the information from its ordering port as a first cut at ordering the candidates, and then its examples port to complete the ordering (again via its information-theoretic measure). KBANN uses its examples port to determine the direction in which to descend the gradient in its greedy search (via backpropagation) and its learning-rate port to determine the size of the steps taken in that direction.

Although the problem-space framework is still in a very preliminary stage of development, one insight already revealed by this analysis is that, though all four of the knowledge-intensive algorithms studied here use EBL-like domain theories, they use them in three qualitatively different ways. Two of the algorithms – KBANN and IVSM – trust their domain theories enough to use them to directly affect the space of candidate hypotheses, though they do this in different fashions. KBANN uses the domain theory to specify the initial space of candidate hypotheses (that is, the network structure). In contrast, IVSM uses the domain theory (along with examples) to constrain the space of candidate hypotheses that was earlier generated from information provided to its partial-order port. FOCL distrusts its domain theory sufficiently to allow it to affect only the search strategy; that is, the domain theory is used only to order the search for a hypothesis, and never to prune the space. It thus gets less constraint from its domain theory, but is also able to recover more gracefully if the theory is wrong. GRENDEL's treatment of the domain theory depends on how the domain theory has been converted into a grammar; GRENDEL can employ either a KBANN-like strategy, in which the theory determines the search space, or a FOCL-like strategy, in which the theory orders the search space. In GRENDEL-FOCL, the variant of GRENDEL discussed above, the domain theory orders the search space.

As work continues on the problem-space framework, the insights derived from it should (hopefully) get both broader and deeper.

Conclusion

We have begun the process of understanding knowledge-intensive induction algorithms by presenting a transformational framework for creating knowledge-intensive methods from knowledge-weak methods, using the framework to analyze four recent algorithms, and deriving from these analyses general implications about the integration of knowledge in induction. We also described a more preliminary problem-space framework that attempts to identify the core functionalities of any learning method and how various learning methods are created by mapping out how knowledge sources can be used to define these functionalities.

Beyond what has already been described, one fundamental insight revealed by these two frameworks, and the analyses they yield of existing knowledge-intensive learners, is a path towards simple yet powerful knowledge-intensive induction algorithms. First, additive ports need to be developed that provide broad languages for the basic functionalities of specifying, constraining and ordering hypothesis spaces. Ideally, such ports and languages should combine, for example, the best aspects of IVSM's version spaces and GRENDEL's grammars, yet still cover all of these basic functionalities. Second, comparable ports need to be developed to allow knowledge to be used in whatever search method is chosen. For greedy methods, this tends to be knowledge about proposing, constraining, and ordering the options at each step. Third, a range of preprocessors need to be created that can translate a wide variety of forms of knowledge into these ports. Ultimately this leads to a direct concern about knowledge extraction, as the preprocessors get closer and closer to the prime sources of knowledge (such as books), and thus raises a variety of additional issues about how and when knowledge is extracted. Ultimately the hope is to complete these two frameworks, fuse them into a single more comprehensive framework, analyze the full space of existing knowledge-intensive induction algorithms, and use the resulting insights to build one or more new algorithms that go significantly beyond the existing ones.

Acknowledgments

This project is partially supported by the National Aeronautics and Space Administration (NASA) under cooperative agreement number NCC 2-538.

References

Buntine, W. (1991). Classifiers: A theoretical and empirical study. *12th International Joint Conference on Artificial Intelligence*. Sydney.

Cohen, W. W. (1992). Compiling prior knowledge into an explicit bias. D. Sleeman & P. Edwards (Ed.), *Machine Learning: Proceedings of the Ninth International Workshop*. Aberdeen.

Hirsh, H. (1990). *Incremental Version Space Merging: A General Framework for Concept*